Scalability of the parallel GCM-T89 code

T. N. Venkatesh, Rajalakshtity Sivaramakrishnan, V. It. Sarasamma and U. N. Sinha

Flosolver Unit, National Aerospace Laboratories, Bangalore 560017, India

The factors governing the scalahility' of parallelization of the GCM-T80 code, operational at NCMRWF, are examined quantitatively. Aspects missed in earlier Indian implementations are discussed. Finally, results of a new hybrid strategy are presented showing significant improvement of parallel efficiency.

NUMERICAL weather prediction is one of the most computationally intensive problems in contemporary science. it is therefore no wonder that weather prediction codes have always been run on the most powerful available computers. An important issue while running these codes on parallel supercomputers is the scalability (i.e. the manner in which the speed-up on the code increases with the number *of processors).*

In an exercise sponsored by the DST's National Centre for Medium-Range Weather Forecasting (NCMRWF) during 1993-1996, many groups in the country parallelized the Global Circulation Model (GCM) T80 code. Basu' has recently reported on the outcome of this project. It is worthwhile to recall that the priority in the project was to *successfully run the operational code* and to *estimate the feasibility of parallel processing for GCM-T80 code'.* From *the point of view of* this objective the project was a success, although this is not apparent from the account in Basi[1]. The issue of scalability was at that time secondary. In Basu`, which is essentially a summary report on the Indian exercise on parallel computing, the issue of scalability is discussed in generic terms like rate of inter-processor communication, intrinsic sequential component of the application code, strategies of para[lelization, etc., but in gross terms records that `the Indian machines, however, have not demonstrated scalability clearly and some more effort in this direction is required'.

The international experience with parallel computing for spectral global circulation model has been quite revealing. The European Centre for Medium-Range Weather Forecasts (ECMWF) does operational forecasts on a distributed memory platform (Fujitsu VPP700) using 4 to 24 Processing Elements (PEs) and has reported the parallel efficiency close to 100%. Even with the full machine (=46 PEs), the parallel efficiency in their case does not fall below 90% (ref. 3). Dent, however, admits that `this has necessitated substantial migration efforts'. Drake *et* al.[5] discuss their parallel implementation and echo similar sentiments in their report. But there are other parallel implementations of the spectral GCM which do not show good scalability. For example, Sela's implementation[6] which is also used for operational forecasts at NMC on a Cray C90 platform (shared memory, vector super computer) gets only a speedup of 3.1 (the parallel efficiency η_p therefore being 77.5%) for 4 processors and 5.5 ($\eta_p = 68.75\%$) for 8 processors.

As the Indian GCM-T80 code running at NCMRWF is very similar to the code running at NMC, it is quite likely that Indian implementations would have similar limitations and these need to be investigated. Achieving scalability is a core issue if Indian meteorological computing is to be efficient from the view of both cost and performance. In this report, a detailed analysis of the scalability of the parallelization of the GCM-T80 code is presented.

Brief description of the model: The GCM-T80 model solves the conservation laws for mass, momentum, energy and moisture. The momentum equations are replaced by the equivalent vorticity η and divergence D equations.

$$\frac{\partial D}{\partial t} = \frac{1}{a\cos^2\phi}\left[\frac{\partial B}{\partial \lambda} - \cos\phi\frac{\partial A}{\partial \phi}\right] - \nabla^2\left[E + \Phi + RT_0 \ln p_*\right], \quad (1)$$

$$\frac{\partial \eta}{\partial t} = \frac{-1}{a\cos^2\phi}\left[\frac{\partial A}{\partial \lambda} + \cos\phi\frac{\partial B}{\partial \phi}\right], \quad (2)$$

where

$$D = \nabla\cdot\mathbf{V};\ \eta = f + \zeta_\kappa;\ \zeta_\kappa = \mathbf{k}\cdot(\nabla\times\mathbf{V});$$

$$A = \eta U + \frac{RT}{a}\cos\phi\left[\frac{\partial \ln p_*}{\partial \phi}\right] + \dot{\sigma}\frac{\partial V}{\partial \sigma} - \cos\phi\, F_\phi;$$

$$B = \eta\,\mathbf{V} - \frac{RT}{a}\left[\frac{\partial \ln p_*}{\partial \phi}\right] - \dot{\sigma}\left[\frac{\partial U}{\partial \sigma}\right] + \cos\phi\, F_\lambda;$$

$$U = u\cos\phi;\ \mathbf{V} = v\cos\phi;$$

$$E = \frac{\mathbf{V}\cdot\mathbf{V}}{2};\ \Phi = \text{geopotential height};$$

f = Coriolis parameter; a = radius of earth;

ϕ = latitude; λ = longitude;

F_ϕ, F_λ = zonal and meridional components of the dissipative force.

∇ = horizontal gradient operator.

In the vertical direction, the σ co-ordinate ($\sigma = p/p_*$, where p is the layer pressure and p_* the surface

pressure) is used. The equation for surface pressure is derived from the continuity equation, and can be written as

$$\frac{\partial \ln p_*}{\partial t} = -\int_0^1 (\nabla \cdot \mathbf{V} + \mathbf{V} \cdot \nabla \ln p_*)\, d\sigma. \quad (3)$$

The energy equation is

$$\frac{\partial T}{\partial t} = -\mathbf{V} \cdot \nabla T + \kappa T \left[\frac{\partial}{\partial T} + \mathbf{V} \cdot \nabla\right] \ln p_* + \frac{H}{C_p} - \pi \dot{\sigma} \frac{\partial}{\partial \sigma}(T/\pi), \quad (4)$$

where $T = \pi\theta$; $\pi = p_*^{\kappa}$; $\kappa = R/C_p$.

The prognostic equation for humidity is

$$\frac{\partial q}{\partial t} = -\mathbf{V} \cdot \nabla q - \dot{\sigma} \frac{\partial q}{\partial \sigma} + S,$$

where S represents the sources and sinks.

The various physical processes in the atmosphere like turbulence, radiation and cloud dynamics are parametrized. To solve this system of equations, the spectral method is used in the T80 because of its superiority over finite difference schemes in terms of accuracy. All the prognostic variables are represented in terms of spherical harmonics.

For example, a variable $F(\phi, \lambda)$ is expanded as

$$F(\phi, \lambda) = \sum_{n=0}^{N} \sum_{l=-n}^{n} F_n^l Y_n^l(\phi, \lambda), \quad (6)$$

where

$$Y_n^l(\phi, \lambda) = P_n^l(\sin\phi)\, e^{il\lambda},$$

$$P_n^l(x) = \frac{1}{2^n n!} \left[\frac{(2n+1)(n-l)!}{2(n+l)!}\right]^{1/2} (1-x^2)^{l/2} \frac{d^{n+l}(x^2-1)^n}{dx^{n+l}}.$$

The number of principal modes is eighty (hence the name T80). This is large enough to ensure that the spatial derivatives in the horizontal direction can be evaluated very accurately. In the vertical direction, finite differences are used. A leap-frog time stepping procedure is used for the time integration[7].

Computational steps: The model computations are carried out in three distinct spaces, namely the two-dimensional global spectral space, the one-dimensional Fourier space (spectral in the azimuthal direction) for each latitude and the two-dimensional latitude/longitude grid space. The Legendre and Fast Fourier Transforms are used to move from one space to another.

The linear computation is done in the spectral space. In other words, the quantities divergence, vorticity, surface pressure, temperature and moisture are expanded in spherical harmonics as in (6) and substituted in the governing equations.

For example, for a particular layer k, the divergence denoted by $D(k)$ is expressed as

$$D(k) = \sum_{n=0}^{N} \sum_{l=-n}^{N} D_n^l(k)\, Y_n^l(\Phi, \lambda),$$

which makes $D(k)$ a vector in the complex vector space of dimension $(N+1)(N+2)/2$. The reduction in dimension is due to the fact that $D(k)$ is a real function. The global divergence vector $\tilde{D}$, covering all the layers, is defined as

$$\tilde{D} = \begin{bmatrix} D(1) \\ D(2) \\ \cdot \\ \cdot \\ \cdot \\ D(l) \end{bmatrix},$$

where l is the number of layers.

With this nomenclature, the governing equations can be written as[7]

$$\frac{\partial \tilde{D}}{\partial t} = \tilde{X} - \nabla^2(\tilde{\Phi} + R\tilde{T}_o \ln p_*) - K_D \nabla^4 \tilde{D}, \quad (7)$$

$$\frac{\partial \tilde{\xi}}{\partial t} = \tilde{W} - K_D \nabla^4 \tilde{\xi}, \quad (8)$$

$$\frac{\partial Q}{\partial t} = Z + S \cdot \tilde{D}, \quad (9)$$

$$\frac{\partial \tilde{T}}{\partial t} = \tilde{Y} + \mathbf{B}\tilde{D} - K_T \nabla^4 \tilde{T}, \quad (10)$$

$$\frac{\partial \tilde{q}}{\partial t} = \tilde{R}_T + \tilde{S}_q - K_T \nabla^4 \tilde{q}. \quad (11)$$

In these equations $\tilde{X}$, $\tilde{Y}$, $\tilde{Z}$, $\tilde{W}$ and $\tilde{R}_T$ represent the nonlinear tendencies. See ref. 7 for details.

A substantial portion of the total computing time is spent in computing the vectors $\tilde{X}$, $\tilde{W}$, $\tilde{Z}$, $\tilde{Y}$ and $\tilde{R}_T$. For the actual time marching, the divergence, temperature and surface pressure equations are discretized in an implicit manner and the vorticity and the moisture equations are treated in a semi-implicit manner. Thus the time marching of vorticity and moisture is done without solving a coupled system of equations, whereas time marching of divergence, temperature and pressure require solving a coupled system of equations. This part of the solution is termed as 'linear dynamics' in Figure 1, which shows the computational flow of the GCM-T80 code. As is evident from Table 1 which

shows the time breakup of the sequential T80 code for a one-day forecast on an SGI Power Challenge machine (90 MHz, R8000), the linear part takes a small fraction of time (around 2%) but for the parallelization to be worthwhile this part cannot be ignored (see Table 2, which shows that even in the limiting case of infinite bandwidth parallelizing efficiency is limited by the sequential component of the algorithm). In our earlier experiments, using the distributed memory architecture machines, parallelization of the linear part had not been possible because the communication overhead offsets any gain made by the parallelization. *In the present work the best features of shared memory and message passing architecture have been exploited to make the communication more effective.* The details are explained later in this article.

Sequential code profile. The profiling, or the measurement of the time taken by the various segments of the code, is the first step in analysing its scalability. Sinha[2] could not profile these fine details due to lack of resources. During the intervening period, powerful profiling tools have been developed in-house and the time breakup of the sequential T80 code for a one-day forecast on an SGI Power Challenge machine (90 MHz R8000) are given in Table 1.

Radiation calculation is not included in this main loop because this is only carried out once in 12/24 hours; hence its effect is secondary in any scheme for effective parallelization.

Analysis of parallelization strategies: As can be seen from Table 1, the subroutines Gloopa and Gloopb, which perform the nonlinear dynamics and physics calculations, account for nearly 98% of the total time. The main computational loop (over all the latitudes) in Gloopa and Gloopb take 95.27% of the total time. Therefore the natural strategy to adopt in parallelization appears to be to start with the latitude loops. This is the strategy which has been used by all the groups in the Indian exercise, and has been a key factor in blunting effectiveness of their parallel implementations. We show later that because of the sequential portion of 4.74% (100–95.26), the parallelization efficiency *decreases sharply as the number of processors increases*. This is the dreaded Amdahl phenomenon[8], particularly detrimental to the effective parallelization of a large code.

Calculation of parallel efficiency: The estimation of the maximum possible parallel efficiency for a given parallelization strategy is calculated as follows. Let the time taken for a sequential run be 100, and let T_s be the time taken for the fraction of the code which is not parallelized. When the number of processors is N, the time taken in the parallel mode (T_p), assuming that the communication bandwidth is infinite, is

$$T_p = T_s + \frac{100 - T_s}{N}.$$

Note that the parallel efficiency is

$$\eta_p = \frac{100}{NT_p}.$$

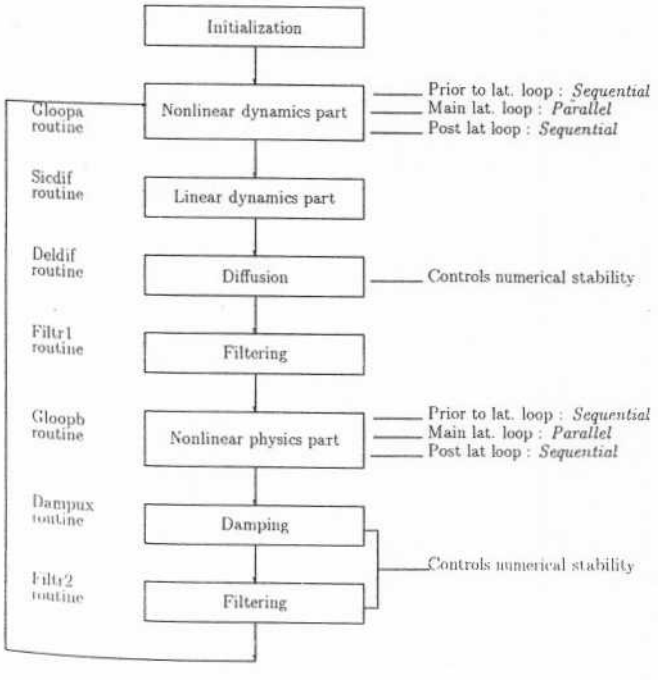

Figure 1. Computational flow of the GCM T-80 code.

Table 1. Profile of the sequential GCM-T80 code for a one-day forecast on SGI Power Challenge machine (90 MHz, R8000)

Subroutines in main computational loop		One day forecast
Gloopa: (nonlinear dynamics)		
	Prior to lat. loop (Dztouv)	27816 (0.87)
	Main lat. loop	1357629 (42.50)
	Post lat. loop	797 (0.03)
Gloopa total time:		1386242 (43.40)
Gloopb: (nonlinear physics)		
	Prior to lat. loop (Dztouv)	24968 (0.78)
	Main lat. loop	1685427 (52.76)
	Post lat. loop	3766 (0.12)
Gloopb total time:		1744029 (54.60)
Sicdif:		46971 (1.47)
Filtr1:		6801 (0.21)
Deldif:		4250 (0.13)
Filtr2:		4119 (0.13)
Dampux:		1567 (0.05)
Others:		156 (0.01)
Main loop total time:		3194135 (100)

Time in milliseconds; Figures in parentheses indicate percentage.

Table 2. Comparison of maximum theoretical and achieved efficiencies

No. of processors	Max. theoretical efficiency		Efficiency achieved*	
	Case A	Case B	Case A	Case B
1	100.0	100.0	100.0	100.0
2	95.5	99.5	93.4	96.1
4	87.6	98.6	81.4	89.0
8	75.1	96.9	–	–
16	58.5	93.6	–	–
32	40.5	87.5	–	–
64	25.2	82.4	–	–

*On a 4 processor SGI Power Challenge (90 MHz, R8000, 512 MB total memory).

Let us now examine two parallelization strategies.

Case A: Only the latitude loops of Gloopa and Gloopb are parallelized. *Sequential part* = 4.74%
Case B: The latitude loops of Gloopa, Gloopb and the subroutines Sicdif, Impadj, Dztouv, Filtr1 and Filtr2 are all parallelized. *Sequential part* = 0.34%

In case A, the sequential part is 4.74% and this leads to a drastic decrease in the parallel efficiency when the number of processors is greater than 4 (Table 2). But in case B, the calculations in the routines other than Gloopa and Gloopb are also parallelized. Since the calculations in these routines are all in the spectral domain and also linear, these routines are easy to parallelize, *but restricting the message passing/communication overhead to a reasonable limit, is much harder.* Following this reasoning, we have actually adopted the strategy in case B on a 4 processor SGI power challenge.

To make the parallelization feasible, shared memory segments were created using UNIX routines supporting shared memory. The program was modified and the overhead was brought down to a level in which any additional memory assignments or cache refreshing gets eliminated. This permits effective parallelization of 4% of the otherwise sequential code and in a sense reflects the ideal combination of distributed and shared programming. That such an hybrid approach is going to be the order of the day is supported by the summary by Keyes *et al.*[9], where they observe that 'Driven by ASCI, large-scale systems will be of distributed-shared memory (DSM) type: shared in local clusters on a node, with the nodes connected by a fast network'. In the present case, the hybridization occurs inside a cluster.

Once all these routines (Sicdif, Filtr1, Filtr2, Dztouv and Impadj) are parallelized, the sequential part (T_s) reduces to 0.34%, and this ensures much better efficiencies when compared to case A. As can be seen from Table 2, with 4 processors, efficiency goes up from 81.4 to 89%, promising much better scalability. The programming details will be published elsewhere.

Conclusions: In the GCM-T80 code, parallelizing only the physical domain latitude loops results in relatively poor efficiencies when the number of processors is greater than 4. To get higher efficiencies, it is necessary to parallelize the linear calculations also in the spectral domain. As a matter of fact, better efficiency has been achieved once the linear calculation in the spectral space is also parallelized (case B), as can be seen from Table 2.

The present study has demonstrated that for viable parallel meteorological computing on more than about 4 processors, the parallelization of linear calculations in the spectral domain is a must. Its implementation demands new ideas together with a better understanding of hardware, system software and application codes. The pursuit of this approach could well trigger a new chapter in parallel computing for meteorological applications in India.

1. Basu, B. K., *Curr. Sri.*, 1998, 74, 508-516.
2. Sinha, U. *N. et rrL, Carr. Sri.*, 1994, 67, 178-184.
3. Dent, D. and Mozdzynski. G., ECMWF Operational Forecasting on a Distributed Memory Platfomt: Forecast Model, Proceedings of the Seventh ECMWF Workshop on the Use of Parallel Processors in Meteorology, 2-6 November 1996.
4. Dent, D., ECMWF Operational Forecasting on a Distributed Memory Platform: General Overview, Proceedings of the Seventh ECMWF Workshop on the Use of Paralel Processors in Meteorology, 2-6 November 1996.
5. Drake, I., Foster, 1., Michalakes, J., Toonen, B. and Worley, P., *I Parallel* Cornpur., 1995, 21, 1571-1591-
6. Seta, J. G., J. *Parallel* Computing, 1995, 21, 1639--1654.
7. Kalnay, E. *et* al., Documentation of the Research version of the NMC Medium Range forecast model, January 1988.
8. Amdahl, G., The validity of the single processor approach to achieving large scale computing capabilities, AFIPS Conference Proceedings, 1967, vol. 30, pp. 483485-
9. David E. Keyes, Dincsh K. Kaushik and Barry F. Smith, Prospects for CFD on Petaflops Systems, NASAICR-97.206279, December 1997.

ACKNOWLEDGEMENTS. We thank Dr T. S. Prahlad, Director, NAL for his encouragement and support throughout this project. We ackrowledge gratefully the discussions, suggestions and editorial help from Prof. R. Narasimha, Director, NIAS and Dr Srinivas Bhogle, Scientist, NAL.

Received 27 May 1998; accepted 7 August 1998